# Experience in Using PCs for Accelerator Instrumentation

W. BLOKLAND
Fermi National Accelerator Laboratory*

## ABSTRACT

The Accelerator Division's Instrumentation Group has been using the graphical programming language, LabVIEW®, on a Macintosh computer for four years. The desktop computer controls data-acquisition units such as GPIB scopes or VME digitizers and communicates with the accelerator control system. The group has successfully implemented a variety of instruments, ranging from bunch intensity monitors to flying wires for transverse sigma measurements. This paper describes our experience in regards to network communication, development, applications, upgrade process, and reliability.

## INTRODUCTION

Four years ago the idea to use a non-embedded computer as basis for accelerator instrumentation was received as quite controversial. An embedded computer with a real-time operating system was (and still is) regarded as a serious work horse while a PC, especially the Macintosh with all its graphics, was regarded as a toy, at most usable for text editing but certainly not appropriate for data-acquisition. The concerns conveyed were about performance and reliability. Can you run an application fast enough on a desktop OS? Is the OS stable enough? Don't you need a real-time OS to do data-acquisition? Is a desktop suitable for industrial use? How is the EMI shielding? How about the hard-disk reliability? Would the computer get stolen? Can you interface to the data-acquiring hardware?

In our group, there were no experts to do the C programming for the application and networking hookup as required for the embedded real-time computers. The experts needed to be borrowed from other groups and might or might not be available depending on their work load. We also wanted to start using as much off-the-shelf hardware and software as possible to save time and money. The embedded systems used at Fermilab did not have the wide range and low cost of commercially available hardware and software that was available for the PC. In addition, the cost of the enclosing crate is unnecessary if no crate modules are used but instead, for example, GPIB instruments.

The type of instrumentation systems we planned to install would have an operation cycle that starts by setting up the hardware, then optionally waits for a trigger, acquires, retrieves and analyzes the data, updates the results, and, if needed, logs the results to a file. This cycle would be around a 1 Hz or slower. An example of this is a system that measures the beam's emittance by flying a wire through it. The only real-time requirements relate to the sampling of the signals, and this is performed by the instrument, not by the computer. As such, there was no requirement for a real-time OS.

On the other side, LabVIEW[1] (at the time available for the Mac only and the desktop of choice in the division) offered an integrated development envi-

---

* Operated by the Universities Research Association under contract with the U.S. Department of Energy.

ronment. The graphical displays and analysis libraries help visualize and develop the data analysis. The available drivers and hardware for various types of instrumentation, e.g., GPIB, CAMAC, VME/VXI, provides for data-acquisition. Adopting LabVIEW allows us to buy most of the hardware and significant parts of the software. As LabVIEW uses the graphical language G, the programming is easier and can be done by the people in our group who do not have a strong background in programming. While LabVIEW does have a learning curve, you do end up in a very complete desktop development environment for applications ranging from on-line instrumentation to modeling.[2]

Therefore, given our needs and requirements, a PC (Macintosh) with LabVIEW offered our group so many advantages that it was worth trying even considering the voiced concerns.

Our first project using LabVIEW initially employed an VXI embedded Macintosh[3], saving us space and guaranteeing good EMI shielding. However, difficulties in adding hardware, e.g., tokenring cards, and lack of upgrade options, forced us to switch to a regular desktop computer with a MXI interface connected to the VXI crate. We found this a much better solution as we did not experience any problems with EMI or durability/reliability, but gained extendibility and lowered the cost. Figure 1 shows a typical configuration of the desktop computer (a Macintosh) connected to a VXI crate or GPIB scope. The desktop running LabVIEW will acquire the data, analyze it, and communicate the results to a console or datalogger.

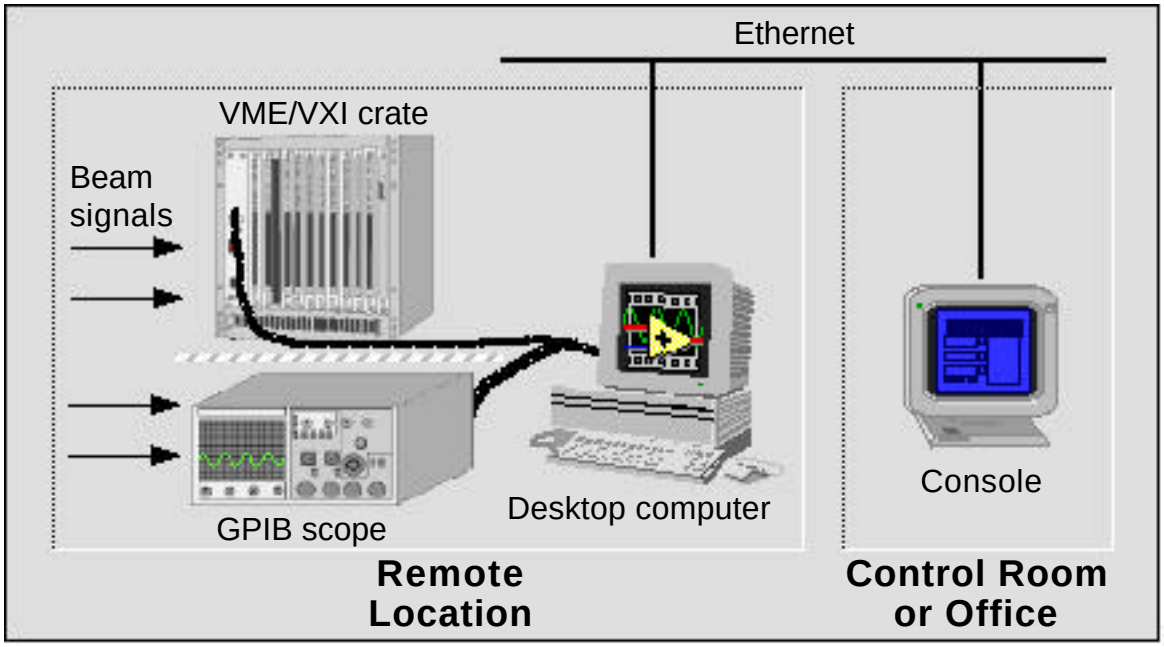


Figure 1. Configuration of Instrumentation.

## NETWORKING

### Introduction

The communication protocol for the Fermilab Accelerator is Acnet. While it is possible to display data on consoles using other protocols, most utilities, like the datalogger and parameter page, rely on the Acnet protocol. Therefore any

instrumentation system must be able to communicate using Acnet. The total of all Acnet functions is quite extensive and would be time consuming to implement. Instead, we chose to implement only those features that were needed, with the intention of adding functions in the future as needed.[4]

## Requirements

The end-user of the data is either an operator or a program like a datalogger. The operator must be able to control the system and retrieve the results. This requires setting and reading capabilities with a desired response time of a fraction (1/5) of a second. But a fixed and guaranteed response time is not needed as the data will not be part of a feedback loop.

Because there are many consoles, multiple requests for data must be handled. A typical load would be several outstanding requests with update rates of about 1 Hz. The possibility of temporarily too high loads are handled by replying standard Acnet errors, indicating lack of resources.

As for the implementation of the interface, the details, like the its programming language or protocol, should be hidden for the application developers so that they do not have to be Acnet nor programming experts to set up an instrumentation system.[2]

## Implementation and Performance

Over the years, the implementation of the interface has been revised three times. The initial version had most Acnet functions run independently of the computer on a smart tokenring card (MCP, Macintosh Coprocessor Platform) with its own OS (A/ROSE, Apple Real-time Operating System Environment). This setup resulted in a fast response time for the Acnet interface and no load on the main processor. As the lab started switching to ethernet and A/ROSE was orphaned by Apple, the Acnet functions were implemented as interrupt driven routines using UDP over ethernet and running on the main processor. This also worked well, delivering fast responses which could only be delayed if interrupt was turned off (e.g., floppy disk access). Because processors became faster, the performance of the single processor setup was actually higher than the dual processor setup. Even faster processors available (e.g., PowerPC) made possible a version completely written in G and as such portable among all LabVIEW platforms. This gave us a wider choice of platforms and independence from the Macintosh. This version does have its drawbacks on a non-preemptive OS, as delays in the message response are unpredictable with other applications running and depend on the load on the processor.

Table 1 shows some of the measured performances of the different interface versions on different machines. The performance is measured as the maximum number of replies (includes data access) to already issued and processed data requests per second and the fastest measured response time (includes processing and data access) for a single shot request message. For the Pentium machine, a very high rate of replies per second is shown with a rather slow response time to a single shot request. We think this has to do with the TCP/IP implementation or OS settings, as we know that the actual processing time of the request is less than 1 msec. All the listed performance numbers satisfy our requirements.

| Interface Version | Computer | Replies per sec | Response (msec) |
|---|---|---|---|
| C on MCP | MCP (68000@10 MHz) | 200 | 11 |
| C on Mac | Mac IIci (68030@30) | 200 | 11 |
| LabVIEW on Mac | PMac 7100 (601@66) | 80 | 13 |
| LabVIEW on PC | PC (P5 @133) | 450 | 20 |

Table 1. Performance of the Acnet Interface.

An additional interface, TCPORT enables the LabVIEW system to read and set other front-ends. The TCPORT interface was written in LabVIEW's G, so it would be portable to other LabVIEW platforms (WindowsNT, HP-UX, Solaris) as well.

## DEVELOPMENT

As the experience with the user-friendliness of the then available embedded systems was quite bad, much of the customization of the LabVIEW programming environment has been focused towards user-friendliness. A template is available to drop one's application into to obtain network connectivity, while the network device registration is done through a spreadsheet table, and read and write access to a device is a simple function call. We found that adding the networking capability to an application was a matter of hours.

We also share as much LabVIEW code as possible to shorten development time. We share the hardware drivers and analysis routines which are mostly application independent and thus easy to share, but also some user-interface and file access functions by writing these as application independent as possible and use these as templates to be customized for each application.

Additional tools[4] are available to generate documentation (WWW), view the program hierarchy, and test the values of the devices over the network.

Because of all the functionality included in LabVIEW, you can set up a working prototype extremely fast, much faster than with the typical C-based environment. This is very nice to do a proof of principle and to see results early on. Finishing up the last 10% of the project still takes about 90% of the time.

## APPLICATIONS

Table 2 shows some of the LabVIEW applications in the Accelerator Division. The applications are typed by

1) Operational: Beam measurements for use by operators, [3,5,6,7]
2) Study: Temporary setups for studies,
3) Alarm: Notification of abnormal conditions,
4) Prototyping: Quick prototyping of a system,[8]
5) Lab: Automated test setups on the lab bench,
6) Display: Remote display of data possible over WWW, and
7) Modeling: Modeling of system behavior.

| APPLICATION | USE | TYPE | TRIGGER | DATA SIZE | ANALYSIS | INSTRUMENT | COMMERCIAL | RESPONSE TIME |
|---|---|---|---|---|---|---|---|---|
| Beamline Tuner | Injection tuning of steering magnets | Oper | Injection Tevatron | 5 kb | N-L damped oscill. fit | VXI digitizer | yes* | 5 seconds |
| Synchrotron Light Monitor | Transverse emit-tance measurement | Oper | Cycle | 100 kb | 2-D non-lin gaussian fit | framegrabber/ HV supply | yes* | 0.3 s/bunch |
| D0 Collision Point Monitor | D0 Collision point determination | Oper | Cycle | 10 kb | Rectifying integrator | GPIB scope | yes* | 90 sec^ |
| B0 Collision Point Monitor | B0 Collision point determination | Oper | Cycle | 10 kb | Rectifying integrator | GPIB scope | yes* | 75 sec^ |
| Booster Ion Profile Mon. | Turn by turn transverse emittance | Oper | Injection Booster | 2.5 Mb | Non-linear gaussian fit | VME digitizers | yes* | 40 s/20k turns |
| MR Ion Profile Mon. | Turn by turn transverse emittance | Oper | Injection MR | 8 Mb | Non-linear gaussian fit | VME digitizers | yes* | 11 s/65k turns |
| Sampled Bunch Display | Long. emittance & int in MR/Tev | Oper | Cycle | 10 kb | Integration & moments | GPIB scope | yes* | 0.5 s/1020 bunches |
| Electron Cool-ing Fly. Wires | Trans. profile display | Oper | User | 15 kb | Plot | VME digitizers | yes* | 15 sec/2 wires |
| Accumulator Flying Wires | Trans. emittance measurement | Oper | User | 20 kb | Non-linear gaussian fit | VME digitizers | yes* | 20 sec/2 wires |
| Tevatron Flying Wires | Trans. emittance measurement | Oper | Superviso-ry program | 100 kb | Non-linear gaussian fit | VME digitzrs Motor cntr. | yes* | 3 sec/72 bunches |
| Gas Jet# | Control of hydrogen over gas jet target | Oper | Cycle | 10 kb | NA | GPIB, serial,PLC | yes* | 2 sec |
| MR Tune monitor | Tune measurements | Study | Injection | 1 kb | Fourier transform | GPIB signal analyzer | yes* | 5 sec |
| MR coupled Bunch | Coupled bunch mode meaurements | Study | User | 10 kb | Two-point correlation | GPIB digitizer | yes* | 15 sec |
| MR coupling | Transverse coupling | Study | User | 10 kb | Cross correlation | GPIB signal analyzer | yes* | 10 sec |
| MR kicker profile | Kicker profile | Study | Injection | 1 kb | Waterfall plot | VXI digitizers | yes* | 5 sec |
| Accumulator Beamloss | Pager Alarm when beam drops | Alarm | Cycle | 1 kb | Comparison | Modem | yes | 5 sec |
| Digital Damper | Damping of coupled bunch modes | Proto | Initiali-zation | 1 kb | NA | in-house VXI modules | no | inits hardware |
| High voltage power supply | Testing HV supply | Lab | User | 10 kb | Plot | GPIB scope | yes | 10 sec |
| Beam Position Modules | Testing of RF modules | Lab | User | 2 kb | Signal response | GPIB sig gen & scope | yes | 30 sec |
| Beam Position Detector | Position/Signal dependency detector | Lab | User | 10 kb | Table | GPIB Anal. Nubus Motor | yes | 30 min |
| MECAR | display, filter download | Display | User | 10 kb | Filter calculation | NA | NA | 1-5 sec |
| WWW Device Page | Display (ascii) of any device for WWW | Display | User | <1 kb | NA | NA | | < 1 sec |
| Device Page | Display of values of any device | Display | User | < 10kb | NA | NA | NA | < 1 sec |
| On-line SBD | 3-D extraction graphics for WWW | Display | 60 seconds | 10kb | NA | NA | NA | 60 s (spill cycle) |
| Bunch Spacing | Bunch arrival times at detector | Model | User | 10 kb | Addition of delays | NA | NA | < 1 sec |
| Sampled Bunch simulation | Model of signal propagation | Model | User | 10 kb | FFT | NA | NA | < 1 sec |
| * in-house timer cards | # under development | | | | | | | |

Table 2. The list of LabVIEW Applications.

The operation of the application can be triggered by an accelerator event (e.g., beam injection), by the user, by a supervisory program, or the program repeating as fast as possible (cycle). The following two columns list the amount of data (in bytes) that must be acquired and how it is analyzed. The next two columns show the type of hardware and whether it was bought or developed in-house. The last column shows the time it takes to acquire, analyze, and present the data. This time gives a general idea about the time-frame of the applications. The times are not directly comparable because different computers are used (e.g., 68000 versus PowerPC Macintosh). Much of the hardware for the systems was commercially available; Most often only the hardware decoding the Fermilab timing signals was made in-house. The synchrotron light monitor, for example, is composed of all commercial hardware except the timing card: CCD camera's, framegrabbers and its software, CAMAC high voltage power supply, GPIB to CAMAC interface, GPIB voltage control, and GPIB interface card[3]. The Sampled Bunch Display[6] uses a GPIB scope and in-house CAMAC timer card. We use Nubus motor controller cards with included software for the implementation of the Flying Wires. We also use video cards to add the graphics display of the Mac to the Fermilab video channels.

Besides buying products related to the data-acquisition, we also buy products like a remote control program for diagnostic access and an installer utility to simplify installation of network software.

We found that given the available hardware support of LabVIEW and third party vendors, we could buy almost all of our hardware and, in many cases software packages were available as well, allowing us to efficiently implement our projects within the LabVIEW platform.

## UPGRADE PROCESS

As the accelerator complex is upgraded for each run, our instrumentation systems are often asked to analyze more data faster. Typically, most of an application's time is spent in analyzing the data. Because several systems are using the same analysis, optimizing this algorithm, even though labor-intensive, is an effective way to increase speed. To increase speed we can also buy faster processors. Because the computers we use are regular desktop computers, they are inexpensive compared to embedded real-time computers, and the innovation cycle is very short because of the high volume and intense competition. Over the years, we have found that we could buy for about $3000 at least a factor of 2 in overall speed per year. The desktop computers can also be assigned for office use, server, and lab test setups. This results in a large pool of computers which are used for spares, replacements, and upgrades. The flexibility of the larger pool of computers has enabled us to immediately meet unexpected user requests for studies or increased performance without first having to purchase new computers, but by reassigning computers.

The major upgrades we have gone through with the Macintosh are the switches from 680x0 processors to PPC 60x processors and from Nubus architecture to PCI architecture. The switch to PPC was easy for the LabVIEW developers as the programs could simply be loaded and would automatically recompile. The C-based interface required several weeks to be ported to PowerPC code. Overall we had very few problems with the PPC conversion. Most of the trouble occurred with the PCI Macintoshes, which initially had an unstable OS and poor performing drivers. Giving the market some time to get rid of the bugs in new hardware or software can save quite a lot of your own time.

Table 2, being an updated version[2], reflects the upgrades of the Sampled Bunch Display[6], Ion Profile Monitor[7], Tevatron Flying Wires, and Synchrotron Light Monitor[5]. The SBD has been upgraded by using a better scope with much faster GPIB transfers and larger record lengths (factor of 20) and a faster computer (factor of 6) to need only 0.5 seconds for determining 1020 bunch intensities in fixed target mode with minimal program changes. The IPM switched from a 68k Nubus Mac to a PCI PowerMac, increasing its speed by a factor of 10. It can now transfer 8 Mbytes of data representing 65000 turns (full Main Ring cycle) of 64 channels of ion profiles, decimate the data and fit a gaussian function with linear offset to 200 profiles, and present this in a 3D graph in about 11 seconds. The TFW switched from a Quadra 950 to a PowerMac 8100 and, together with the faster analysis routines obtained a tenfold speed increase to fly three wires and analyze 72 profiles of 80 points in a total of 3 seconds. The SLM added a $500 accelerator card with a 66 MHz PPC 601 to its Quadra 950 and got a threefold speed increase.

## RELIABILITY

In general, we found that the combination of MacOS and LabVIEW provides a stable application. On completion of the development we would have a reliable system. A leading cause of down-time is a site-wide power outage. As such our longest running time is about 180 days.

During development we do encounter unexplained crashes, probably due to interaction of applications. Here the Mac has a definite disadvantage to the embedded real-time computer with a very well defined OS. The MacOS can become rather unwieldy if many extensions and utilities are installed, and one program's bug can crash another program. To prevent crashes, a Mac is set up as a bare bones machine with only those programs installed that we really need.

Within LabVIEW, you do not deal directly with pointers and memory allocation and thus avoid certain bugs. The trade-off is that you, as a developer, have much less control over the memory allocation. One danger of this is that an application that processes large amounts of data, e.g., the IPM, can run out of memory as copies of the large buffers are being made automatically. We found here that we needed to be very careful with the program design to minimize memory usage, and we needed to add RAM to the Mac, what would not have been needed for a C-based programming environment.

### FUTURE

As the networking interface is now available for other LabVIEW platforms as well, we are testing the reliability and performance of other platforms, in particular WindowsNT because of its preemptive and memory protected OS. A switch from MacOS to WindowsNT is currently not necessary for us, as everything we want to do can be done with MacOS. However, new developments, either market driven or switch of desktop computer in the division, might require us to switch in the future.

Another step contemplated is to further simplify and generalize the communication interface by using an intermediate node that translates Acnet requests into a very simple protocol that only mirrors memory and uses only single shot settings to update data on either side. This simplified protocol is

then very easy to implement in any programming language and would make porting to any hardware or software platform very easy, as little or no OS specific functions are needed for the interface. The simple protocol would be on top of TCP/IP for generality and ease of networking.

## SUMMARY

All in all we are very pleased with using LabVIEW as our development and application environment. It enabled people, from technicians to hardware engineers, in our group to complete many applications in a time frame that would not have been possible using more traditional tools, such as C. We were able to use existing hardware and buy additional components of the system, whether it be software or hardware, to reduce development time. We had excellent results with the use of desktop computers and had the extra benefits of reduced cost, flexibility, software and hardware availability, and upgrade paths. LabVIEW allowed us to minimize the need for specialist programmers and maximize the number of people that can work on a instrumentation system, whether is for operational or lab bench purposes.

## REFERENCES


[1] LabVIEW®. Made by National Instruments.
[2] W. Blokland, "A LabVIEW-based Accelerator Instrumentation Platform", EPAC94, London, GB, 1994, pp. 1527-1529.
[3] W. Blokland, "A VXI/LabVIEW-based Beamline Tuner", PAC'93, Washington, USA, 1993.
[4] W. Blokland, "Integrating the commercial software package LabVIEW with Fermilab's Accelerator Control NETwork", ICALEPCS, Chicago, 1995.
[5] A. A. Hahn and P. Hurh, "Results From An Imaging Beam Monitor In The Tevatron Using Synchrotron Light", HEACC'92, Hamburg, Germany, July 1992, pp. 248-250.
[6] E. Barsotti, "A longitudinal Bunch Monitoring System using LabVIEW and high-speed oscilloscopes", 1994 Accelerator Instrumentation Workshop, Vancouver, Canada, 1994, pp. 466-472.
[7] J. Zagel, "Booster Ion Profile Monitor using LabVIEW", 1994 Accelerator Instrumentation Workshop, Vancouver, Canada, 1994, pp. 384-390.
[8] J. Steimel, "Fast Digital Damper for the Fermilab Booster", PAC95, 1995.